EECS 4330/5330/7330 Class notes

Draft

These lecture notes are provided for your convenience. It is no substitute for the text book/lectures and may contain typo mistakes which would be corrected in the class.

Image transformation

2-D Transformations

$$P = [x \quad y]$$

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$x^* = ax + cy$$
$$y^* = bx + dy$$

Where x* and y* are the new coordinates

Some simple examples

- Identity Matrix (no change)

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$x^* = x$$
$$y^* = y$$

- Matrix for a scaled x-axis

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}$$

$$x^* = ax$$
$$y^* = y$$

- Reflection about y-axis

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$x^* = -x$$
$$y^* = y$$

- Shearing affect

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$$

$$x^* = x$$
$$y^* = bx + y$$

In this case we can't obtain a "translation" , i.e.,

$$[x^* \quad y^*] = [x \quad y]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

where we need,

$$x^* = x + m$$
$$y^* = y + n$$

This transformation can only be obtained by expanding both matrixes into a homogeneous coordinate system by adding the 3rd Dimension. This gives us the following equations.

$$[x \quad y \quad 1]\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix}$$
$$x^* = x + m$$
$$y^* = y + n$$

The problem with this equation set is that we want the transformation matrix to be square (i.e. 3x3) so we can obtain its inverse. This will allow us to undo the transformation. The expanded matrix is shown below.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

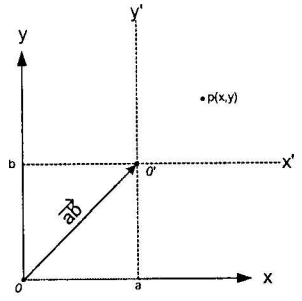
We must have a homogenous coordinate system to perform operations uniformly. To obtain the homogeneous coordinate system, we simply add an additional dimension. So for a normal 3-Dimensional object in the (x,y,z) coordinate system (Cartesian), we get 3 + 1 = 4.
Given a point (x,y) we can perform its expansion to the homogeneous coordinate system as shown below.

$$\begin{bmatrix} x \\ y \end{bmatrix}$$
$$\begin{bmatrix} kx \\ ky \\ k \end{bmatrix}$$

We simply add an extra dimension and then scale each coordinate by k. The k value in many normal cases is 1.

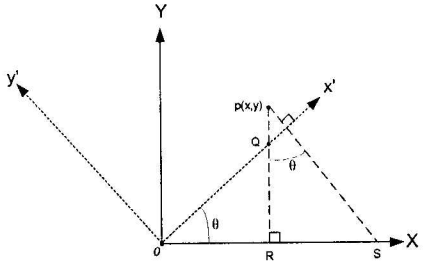
Imaging Geometry

- Translation by a,b



$$x' = x - a$$
$$y' = y - b$$

- Rotation by θ



$$x' = OS\cos\theta \qquad y' = (y - QR)\cos\theta$$
$$x' = (OR + RS)\cos\theta \qquad y' = (y - x\tan\theta)\cos\theta$$
$$x' = (x + y\tan\theta)\cos\theta \qquad y' = y\cos\theta - x\sin\theta$$
$$x' = x\cos\theta + y\sin\theta \qquad y' = -x\sin\theta + y\cos\theta$$

$$[x' \quad y'] = [x \quad y]\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

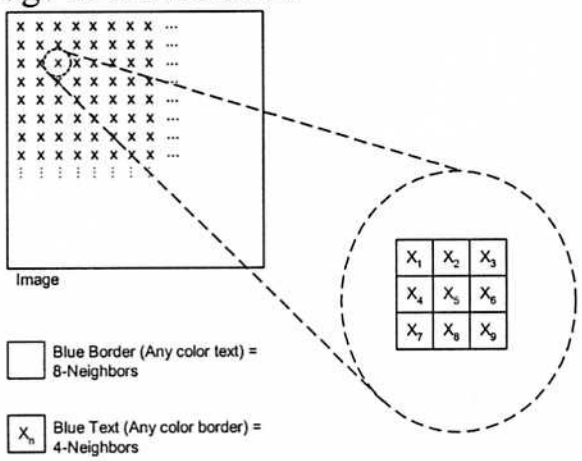
- Scaling

$$x' = rx$$
$$y' = sy$$

where r, s are scaling factors

In the following image, x's represent pixel intensity values

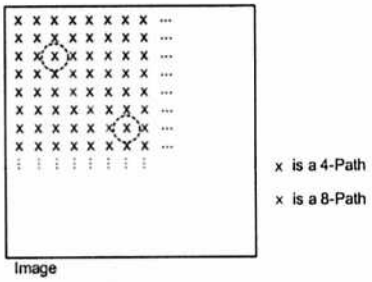
- Intensity values are normally in the range of 0-255
- There are two sets of neighbors, 8-Neighbors and 4-Neighbors for any point in the image as shown below



- ***4-Path*** is a sequence of points (P_0, P_1, P_2, P_3, …, P_n) where P_i is a 4-Neighbor of P_{i-1}

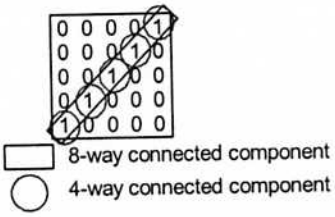
- ***8-Path*** is a sequence of points (P_0, P_1, P_2, P_3, …, P_n) where P_i is a 8-Neighbor of P_{i-1}

- Example



- A connected component is a set of points in which there is a path between any two points in the set. The type of a connected component depends upon the connectivity used (8-way or 4-way connectivity). It all points in a region are connected by a 4-path then it has 4-way connectivity. For 8-way connectivity an 8-path must exist.

- Examples of connected components
    - Note that in 8-way connectivity the "1's" create a single connected component, but in 4-way connectivity there are 5 separate connected components.



Distance between two points

Give two points P(x,y) and Q(u,v), the following equations are used to calculate each type of distance.

Euclidean distance d_e

$$d_e = \sqrt{(x-u)^2 + (y-v)^2}$$

City Block Distance d_4

$$d_4(P,Q) = |x-u| + |y-v|$$

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	x	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

Image (City Block)

Chessboard distance d_8

$$d_8(P,Q) = \max(|x-u|, |y-v|)$$

3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	x	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

Image (Chessboard)

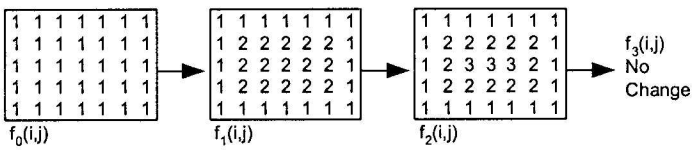
Distance Transform

It is the Point's Distance from the boundary.itis obtained recursively. The process continues until there is no change in f .

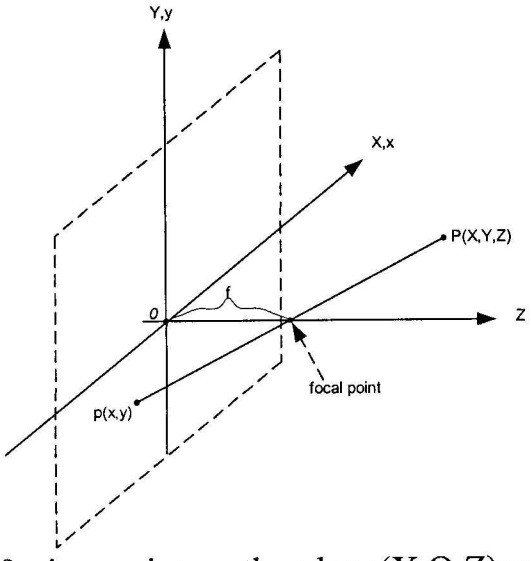
$$\overset{0}{f}(i,j) = f(i,j)$$

$$\overset{m}{f}(i,j) = \overset{0}{f}(i,j) + \min(\overset{(m-1)}{f}(u,v))$$

$$d[(u,v),(i,j)] = 1$$

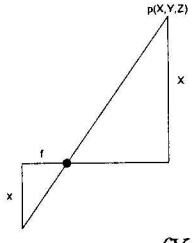


Perspective Transformation



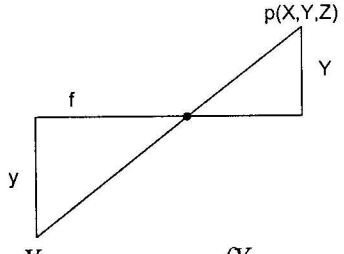
If p is a point on the plane(X,O,Z) with f(focal length), then

In the horizontal plane,



$$\frac{x}{f} = -\frac{X}{Z-f} \rightarrow x = -\frac{fX}{Z-f} \qquad (1)$$

In the vertical plane,



$$\frac{y}{f} = -\frac{Y}{Z-f} \Rightarrow y = -\frac{fY}{Z-f} \qquad (2)$$

Equations 1 & 2 are nonlinear, to get the matrix forms of these transforms we first convert the (x,y,z) coordinates into homogenous world coordinates,

$$w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$w_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix}$$

$$C_h = PW_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} k & X \\ k & Y \\ k & Z \\ k & \end{bmatrix} = \begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{f} + k \end{bmatrix}$$

To move from homogeneous to normal divide by the last coordinate and then drop the last coordinate. You get,

$$\Rightarrow \begin{bmatrix} -\frac{fX}{Z-f} \\ -\frac{fY}{Z-f} \\ -\frac{fZ}{Z-f} \\ 1 \end{bmatrix} \Rightarrow C = \begin{bmatrix} -\frac{fX}{Z-f} \\ -\frac{fY}{Z-f} \\ -\frac{fZ}{Z-f} \end{bmatrix}$$

The following is the X component derivation of the above calculation

$$\frac{kX}{\left(-\frac{kZ}{f}+k\right)} = \frac{X}{\left(1-\frac{Z}{f}\right)} = -\left(\frac{Xf}{Z-f}\right) = -\frac{fZ}{Z-f}$$

Inverse Perspective Transformation

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}$$

3D to 2D transformations cause a loss of data, so the original 3D image can not be reproduced. We can determine the line that each point would be on, but not the "exact" location.

Using the image point C_h we can obtain the line that point would have originally been on by the following procedure.

$$C_h = \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ k \end{bmatrix}$$

$$w_h = P^{-1}C_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ k \end{bmatrix} \Rightarrow w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{fx_0}{f+z} \\ \frac{fy_0}{f+z} \\ \frac{fz}{f+z} \end{bmatrix}$$

$$Z = \frac{fz}{f+z} \Rightarrow fz = Zf + Zz \Rightarrow fz - Zz = Zf \Rightarrow z(f-Z) = Zf \Rightarrow z = \frac{Zf}{f-Z}$$

$$X = \frac{fx_0}{f + \frac{Zf}{f-Z}} = \frac{x_0}{f}(f-Z)$$

$$Y = \frac{fy_0}{f + \frac{Zf}{f-Z}} = \frac{y_0}{f}(f-Z)$$

Image Enhancements

Enhancement – deals with the problem with no knowledge of the source of degradation
Restoration – deals with the problem with some knowledge of the source of degradation
e.g.,

$$g = f * h \overset{fourier}{\Rightarrow} G = F \cdot H \Rightarrow F = H^{-1}G$$

g is the degraded image
f is the original image
h is the degradation
The convolution of f and h produces the degraded image g. To restore the original image, we use the Fourier transforms.

Measures of Fidelity of an Image

1.

$$\iint (f-g)^2 \, dxdy \Rightarrow \sum_{j=1}^{N-1} \sum_{i=1}^{N-1} [f(i,j) - g(i,j)]^2$$

2.

$$\iint |(f-g)| dxdy$$

3.

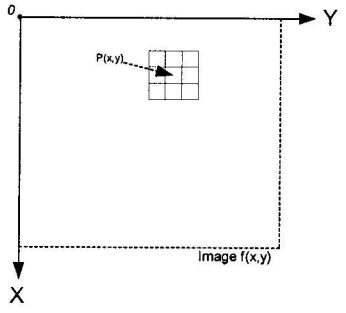
$$\max |f-g|$$

Image Histogram (count of no. of each pixel intensity)

$$probability = \frac{Histogram\_Bar}{Length * Height(TotalPixels)}$$

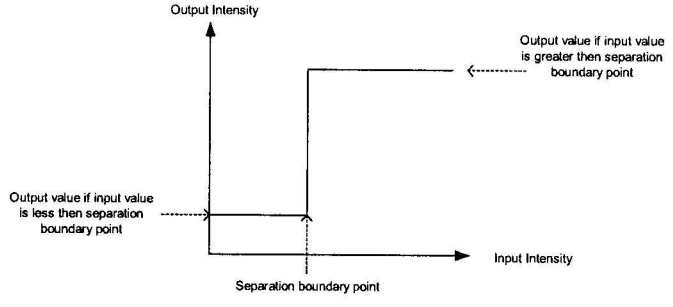
Image Coordinate System

- The following image shows an image coordinate system. The image also shows a 3by3 neighborhood around a point p(x,y) in an image f(x,y).



Thresholding

- In thresholding all intensity values in the image are mapped from their current value to one of two chosen values. Any value greater than the threshold value is mapped to the right horizontal line and vice versa for any value less than the chosen intensity value.



Gray Level Correction

$$g(x,y) = e(x,y) \cdot f(x,y)$$

g(x,y) is the observed image
f(x,y) is the original image
e(x,y) is the error function
C is an image with constant gray level value C.

Calibration Step

$$g_c(x,y) = Ce(x,y) \Rightarrow e(x,y) = \frac{g_c(x,y)}{C}$$

Then

$$f(x,y) = \frac{g(x,y)}{e(x,y)}$$

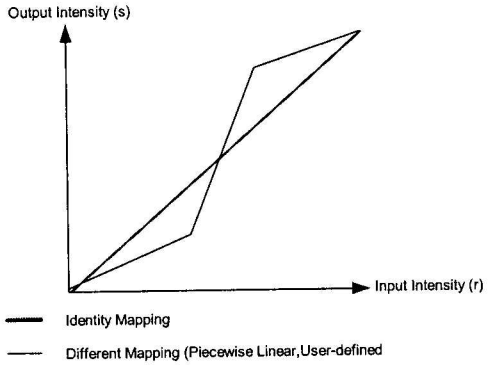
Contrast Stretching

Using relative lengths

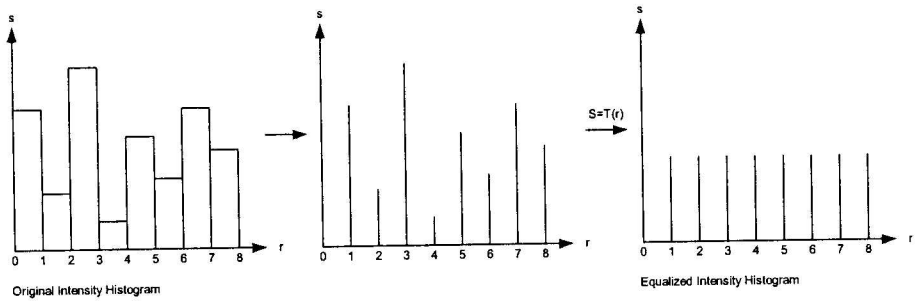
$$\frac{Z - Z_1}{Z_k - Z_1} = \frac{Z' - Z'_1}{Z'_k - Z'_1}$$

$$Z' = \frac{Z'_k - Z'_1}{Z_k - Z_1}(Z - Z_1) + Z'_1$$

Examples of types of contrast stretching functions



r is the input grey intensity level normalized to 1.



$P_r(r)$: probability density function of gray levels "before" transformation
$P_s(s)$: probability density function of gray levels "after" transformation

$$P_s(s) = P_r(r)\left.\frac{dr}{ds}\right|_{r=T^{-1}(s)}$$

$$s = T(r) = \int_0^r P_r(w)dw$$

$$\frac{ds}{dr} = \int_0^r \frac{d}{dr}P(w)dw + P_r(r)\frac{dr}{dr} - P(0)\frac{d0}{dr}$$

Note,

$$G(u) = \int_{a(u)}^{b(u)} H(x,u)dx$$

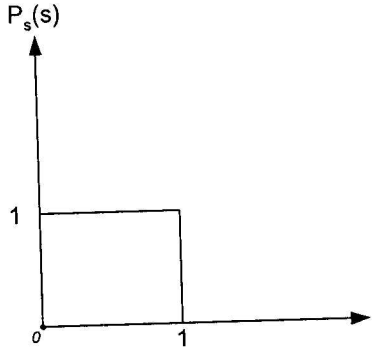
$$\frac{dG(u)}{du} = \int_{a(u)}^{b(u)} \frac{dH}{dx}dx + H(b,u)\frac{db(u)}{du} - H(a,u)\frac{da(u)}{du}$$

Therefore,

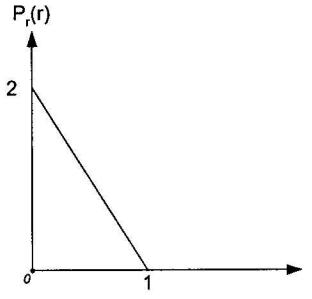
$$P_s(s) = P_r(r)\left.\frac{dr}{ds}\right|_{r=T^{-1}(s)}$$

$$P_s(s) = P_r(r)\left.\frac{1}{P_r(r)}\right|_{r=T^{-1}(s)} = 1$$

This give us the uniform density function



Example:



$$P_r(r) = \begin{cases} -2r+2 & 0 \le r \le 1 \\ 0 & else \end{cases}$$

$$s = T(r) = \int_0 P(w)dw = \int_0 (-2w+2)dw = -r^2 + 2r$$

Actual Integration

$$-2\int_0 wdw + 2\int_0 dw$$

$$-2\frac{w^2}{2}\Big|_0^r + 2\int_0 dw$$

$$-2\frac{w^2}{2}\Big|_0^r + 2w\Big|_0^r = (-r^2 + 0^2) + (2r + 0) = -r^2 + 2r$$

$$s = T(r) = -r^2 + 2r$$
$$r = T^{-1}(s) = 1 \pm \sqrt{1-s}$$

This is done using the quadratic formula as shown below

$$-r^2 + 2r - s = 0$$

$$r^2 - 2r + s = 0$$

$$r = \frac{2 \pm \sqrt{4-4s}}{2} = \frac{2 \pm 2\sqrt{1-s}}{2} = 1 \pm \sqrt{1-s}$$

We take

$$r = 1 - \sqrt{1-s}$$

because r is less than or equal to 1.

$$P_s(s) = P_r(r)\frac{dr}{ds}\Big|_{r=T^{-1}(s)}$$

$$\frac{dr}{ds} = 0 - \frac{-1}{2\sqrt{1-s}} = \frac{1}{2\sqrt{1-s}}$$

Continuing,

$$(-2r+2)\frac{1}{2\sqrt{1-s}}\Big|_{r=1-\sqrt{1-s}}$$

$$\frac{-2(1-\sqrt{1-s})+2}{2\sqrt{1-s}} = \frac{-1+\sqrt{1-s}+1}{\sqrt{1-s}} = \frac{\sqrt{1-s}}{\sqrt{1-s}} = 1$$

In the discrete case, let $P_r(r_k)$ be the probability of the kth gray level,

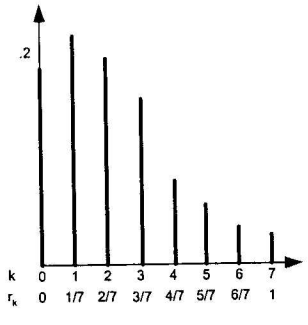
n_k = total number of pixels with level r_k

n = total number of pixels in the image
then,

$$P_r(r_k) = \frac{n_k}{n}$$

$$S_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} P(r_j)$$

Example



r_k	n_k	$P_r(r_k) = \frac{n_k}{n}$
r_0=0	790	.19
r_0=1/7	1023	.25
r_0=2/7	850	.21
r_0=3/7	656	.16
r_0=4/7	329	.08
r_0=5/7	245	.06
r_0=6/7	122	.03
r_0=1	81	.02
Total	4096	

Pixel Calculations

790 $pixels\ map\ to \begin{cases} S_0 = T(r_0) = \sum_{j=0}^{0} P(r_j) = P(r_0) = .19 = \frac{1}{7} \end{cases}$

1023 $pixels\ map\ to \begin{cases} S_1 = T(r_1) = \sum_{j=0}^{1} P(r_j) = P(r_0) + P(r_1) = .19 + .25 = .44 = \frac{3}{7} \end{cases}$

850 $pixels\ map\ to \begin{cases} S_2 = T(r_2) = \sum_{j=0}^{2} P(r_j) = .19 + .25 + .21 = .65 = \frac{5}{7} \end{cases}$

985 $pixels\ map\ to \begin{cases} S_3 = T(r_3) = \sum_{j=0}^{3} P(r_j) = .19 + .25 + .21 + .16 = .81 = \frac{6}{7} \\ S_4 = .81 = \frac{6}{7} \end{cases}$

448 $pixels\ map\ to \begin{cases} S_5 = .95 = 1 \\ S_6 = .98 = 1 \\ S_7 = 1 \end{cases}$

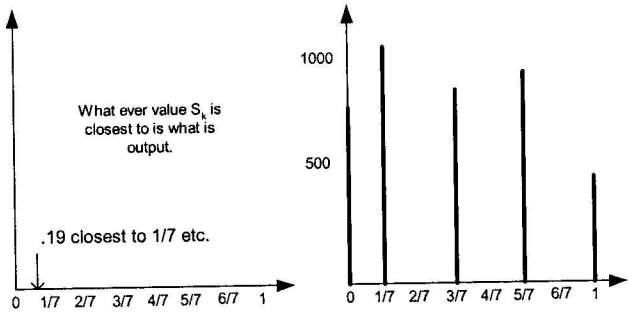
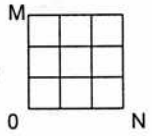


Image Enhancement (continued)

1. Picture Smoothing



For an MxN neighborhood

$f_{M,N} = \frac{1}{|S|}\sum_{(i,j)\in S} f(i,j)$ where S denotes the neighborhood.

Find the $f_{M,N}$ and replace the center point with this new value.

Major problem with picture smoothing is sharp edges get blurred

Weighted Average (Gaussian Smoothing)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \qquad 1-D$$

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{2\pi\sigma^2}\left[e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}}\right] \qquad 2-D$$

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = \frac{1}{4}\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{4^2}\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

The separable operators in Gaussian allows for less computations. Instead of doing nxn=n^2 operations we do 2n operations.

2. Median Filtering

To perform median filter we take a neighborhood such as

10	12	10
15	12	18
25	14	11

and take all the intensity values within the neighborhood and sort them in ascending order: 10, 10, 11, 12, 12, 14, 15, 18, 25. We then take the median value (middle value) and replace the center of the neighborhood with it. In this case the median value is 12 so the center value does not change.

This filtering technique preserves the edges better (less blur).

In doing these filtering operations we want to reduce the noise within the image, but not scale up the values, hence the reason for using "1".

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = 9 * \frac{1}{9}(normalizingfilter) = 1$$

Notes about assignment:
(solution to part 1 & 2)

```
Clear all
Load U:eecs/image/lenna
Colormap(gray(256))
Image(lenna)
x = 1:256;
y = zeros(1, 256);
for i = 1:256;
        for j = 1:256;
                y(lenna(i,j)) = y(lenna(i, j))+1
        end
end
bar(x,y)
title('Intensity Histogram')
Xlabel('Intensity')
Ylabel('Numbers')
```

Percentage of Pixels at intensity b_i

$$P(b_i) = \frac{100 \cdot N(b_i)}{L \cdot S}$$

L is the length of the image
$N(b_i)$ is the number of pixels of intensity b_i
S is width of the image

Percentage of Pixels at *or* below intensity b_i

$$C(b_i) = \sum_{k=1}^{i} P(b_k) = P(b_1) + P(b_2) + \ldots + P(b_i)$$

$$C(b_i) \geq 20\%$$

Also, for (mean 20, variance 50)
X = randn(256) (This is by system default mean 0, variance 1)
In general,
Y = ax+b
$E[ax+b] = \mu_{new}$
$V[ax+b] = (\sigma_{new})^2$

$a\,\mu_{old} + b = \mu_{new}$ → b
$a^2\,(\sigma_{old})^2 = (\sigma_{new})^2$ → $a = \sigma_{new} / \sigma_{old}$

This would give us x = randn(256) and $y = \sqrt{20}\,x + 20$ for a mean 20, variance 20
In our case $\mu_{old} = 0$, $\sigma_{old} = 1$, since we want 2 and 4 plug in for the Variable_{new} values and solve for a and b.

Image Sharpening

Uses differentiation operator

The derivative operator for edge detection needs to be isotropic operator – Equally sensitive to edges of all directions.

$f(i,j)-f(i,j-1)$ detect edges (responds high) in vertical direction.

$f(i,j)-f(i-1,j)$ detect edges (responds high) in horizontal direction.

$$x = x'\cos\theta - y'\sin\theta$$

$$y = x'\sin\theta - y'\cos\theta$$

$$\frac{\partial f}{\partial x'} = \frac{\partial f}{\partial x}\cdot\frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y}\cdot\frac{\partial y}{\partial x'} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

$$\frac{\partial f}{\partial y'} = \frac{\partial f}{\partial x}\cdot\frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y}\cdot\frac{\partial y}{\partial y'} = -\frac{\partial f}{\partial x}\sin\theta + \frac{\partial f}{\partial y}\cos\theta$$

$$\left(-\frac{\partial f}{\partial x}\sin\theta + \frac{\partial f}{\partial y}\cos\theta\right)^2 + \left(\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right)^2 = \left(\frac{\partial f}{\partial y'}\right)^2 + \left(\frac{\partial f}{\partial x'}\right)^2 = \left(\frac{\partial f}{\partial y}\right)^2 + \left(\frac{\partial f}{\partial x}\right)^2$$

i.e.it is isotropic operator.

we define,

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial y}\right)^2 + \left(\frac{\partial f}{\partial x}\right)^2}$$

$$\theta_f = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right)$$

(Magnitude and Direction) of Gradient(f).

In discrete domain,

$$\Delta_x f(i,j) = f(i,j) - f(i-1,j)$$

$$\Delta_y f(i,j) = f(i,j) - f(i,j-1)$$

This set of equations is for finding a derivative at a point (i,j). Therefore, the gradient of f can be found as shown below.

$$|\nabla f| = \sqrt{(\Delta_x f(i,j))^2 + (\Delta_y f(i,j))^2}$$

$$\theta_f = \tan^{-1}\left(\frac{\Delta_y f(i,j)}{\Delta_x f(i,j)}\right)$$

Laplacian

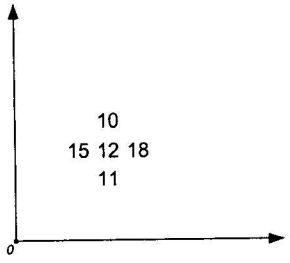
$$|\nabla^2 f| = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\Delta_x^2 f = \Delta_x(\Delta_x f) = \Delta_x f(i+1,j) - \Delta_x f(i,j)$$
$$\Delta_x^2 f = f(i+1,j) - f(i,j) - [f(i,j) - f(i-1,j)]$$
$$\Delta_x^2 f = f(i+1,j) - f(i,j) - f(i,j) + f(i-1,j)$$
$$\Delta_x^2 f = f(i+1,j) - 2f(i,j) + f(i-1,j)$$
$$\Delta_y^2 f = \Delta_y(\Delta_y f) = \Delta_y f(i,j+1) - \Delta_y f(i,j)$$
$$\Delta_y^2 f = f(i,j+1) - f(i,j) - [f(i,j) - f(i,j-1)]$$
$$\Delta_y^2 f = f(i,j+1) - 2f(i,j) + f(i,j-1)$$

Laplacian operator is defined as

$$\nabla^2 f(i,j) = \Delta_x^2 f(i,j) + \Delta_y^2 f(i,j)$$
$$\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j-1) + f(i,j+1) - 4f(i,j)$$

Example



$\Delta_x^2 = 18 + 15 - 24 = 9$

$\Delta_y^2 = 11 + 10 - 24 = -3$

$\nabla^2 f(i,j) = 18 + 11 + 15 + 10 - 4(12) = 6$

$\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j-1) + f(i,j+1) + f(i,j) - 5f(i,j)$

$\nabla^2 f(i,j) = -5\left[f(i,j) - f_{avg}(i,j)\right]$

$\nabla^2 f(i,j) = f(i,j) - f_{avg}(i,j)$

This last equation is called unsharp masking.

Example

f:	0	0	0	1	2	3	4	5	5	5	5	5	5	6	6	6	6	6	6	3	3	3	3
Δ_x^2f:	0	0	1	0	0	0	0	-1	0	0	0	0	1	-1	0	0	0	0	-3	3	0	0	0
f-Δ_x^2f:	0	0	-1	1	2	3	4	6	5	5	5	5	4	7	6	6	6	6	9	0	3	3	3

Note: Subtracting the Laplacian from the original image increases the ramp steepness.

Image Segmentation

Method 1: Thresholding

$$f_t(x,y) = \begin{cases} 1 & if \quad f(x,y) \geq t \\ 0 & if \quad f(x,y) < t \end{cases}$$

Pdf = Probability density function

$f_x(x) = p(X = x)$

Cdf = Cumulative Distribution function

$$F_x(x) = \int_{-\infty}^{x} f_x(x')dx'$$

$$F_x(x) = p(X \le x)$$

Minimum Error Thresholding

- Assume the picture consists of object and background

- Let p(z|1) and p(z|2) be the pdf for the object and the background respectively.

$$p(z) = p(z \mid 1)p(1) + p(z \mid 2)p(2)$$

$$p(A) = \sum_n p(A \mid B_n)p(B_n)$$

$$p_e = p(1)p_{e_1} + p(2)p_{e_2}$$

$$p_{e_1} = p(z \mid 1 > t) = \int_t^{\infty} p(z \mid 1)dz = \int_{-\infty}^{t} p(z \mid 1)dz = 1 - P(t \mid 1)$$

$$p_{e_2} = \int_{-\infty}^{t} p(z \mid 2)dz = P(t \mid 2)$$

$$p_e = p(1) \cdot [1 - P(t \mid 1)] + p(2)P(t \mid 2)$$

- Assume p(1)=θ P(2)=1- θ

$$p_e = \theta \cdot [1 - P(t \mid 1)] + (1 - \theta)P(t \mid 2)$$

- To minimize

$$\frac{\partial p_e}{\partial t} = -\theta \cdot p(t \mid 1) + (1 - \theta)p(t \mid 2) = 0$$

$$(1 - \theta)p(t \mid 2) = \theta \cdot p(t \mid 1)$$

- As an example

$$p(t \mid 1) = \frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}}$$

$$p(t \mid 2) = \frac{1}{\sqrt{2\pi}\cdot\tau} e^{-\frac{1}{2}\frac{(t-\nu)^2}{\tau^2}}$$

$$\ln\left[(1-\theta)\frac{1}{\sqrt{2\pi}\cdot\tau} e^{-\frac{1}{2}\frac{(t-\nu)^2}{\tau^2}}\right] = \ln\left[\theta\frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}}\right]$$

$$-\frac{1}{2}\frac{(t-\nu)^2}{\tau^2} = \ln\frac{\theta\tau}{\sigma(1-\theta)} \cdot -\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}$$

$$\left[-\frac{1}{2}\frac{(t-\nu)^2}{\tau^2} = \ln\frac{\theta\tau}{\sigma(1-\theta)} \cdot -\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}\right] 2\tau^2\sigma^2 = \sigma^2(t-\nu)^2 = 2\tau^2\sigma^2 \ln\frac{\theta\tau}{\sigma(1-\theta)} \cdot -\tau^2(t-\mu)^2$$

$$\tau^2(t-\mu)^2 - \sigma^2(t-\nu)^2 = 2\tau^2\sigma^2 \ln\frac{\theta\tau}{\sigma(1-\theta)}$$

- Lets consider a special case:

$$\theta = \frac{1}{2}, \quad \tau = \sigma$$

gives

$$(t-\mu)^2 = (t-\nu)^2$$

$$t-\mu = \pm(t-\nu)$$

$$t-\mu = -t+\nu$$

$$2t = \mu+\nu$$

$$t = \frac{\mu+\nu}{2}$$

Note that part of the background will become part of the foreground during the thresholding process and vice versa. The question is where to put the threshold values such that this crossover error is minimized.

Iterative Threshold Selection Algorithm

1. Select an initial estimate of the threshold T. A good initial value is the average intensity of the image.
2. Partition the image into two groups, R_1 and R_2, using the threshold T.
3. Calculate the mean gray value μ_1 and μ_2 of the partitions R_1 and R_2.
4. Select a new threshold:

$$T = \frac{1}{2}(\mu_1 + \mu_2)$$

5. Repeat steps 2-4 until the mean values of μ_1 and μ_2 in successive iterations do not change.

Variations of Gradient Operator

$$\sqrt{(\Delta_x f(i,j))^2 + (\Delta_y f(i,j))^2}$$

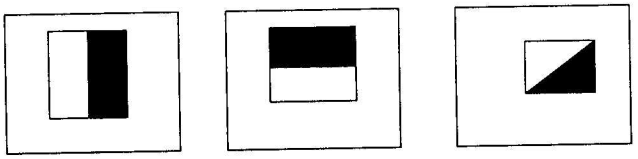
$$|\Delta_x f(i,j)| + |\Delta_y f(i,j)|$$

$$\max(|\Delta_x f(i,j)|, |\Delta_y f(i,j)|)$$

where,

$$\Delta_x f(i,j) = f(i,j) - f(i-1,j)$$

$$\Delta_y f(i,j) = f(i,j) - f(i,j-1)$$



$\Delta_y = 0$ $\Delta_x = 0$ $\Delta_x = \Delta_y$

On the last example (diagonal) the three operators actually give different answers (sqrt(2)|deltax|), 2|deltax| and |deltax| respectively.

Note: Various definitions of norm of a vector

$$x = (x_1, x_2, x_3, \ldots, x_n)$$

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

$$\|x\| = \sum_{i=1}^{n} |x_i|$$

$$\|x\| = \max(|x_i|, i = 1,2,\ldots,n)$$

Some new definitions,

$$\Delta_{2x} f(i,j) = f(i+1,j) - f(i-1,j)$$

$$\Delta_{2y} f(i,j) = f(i,j+1) - f(i,j-1)$$

$$Maskx = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$Masky = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Roberts Operator

$$\Delta_+ f(i,j) = f(i+1, j+1) - f(i,j)$$
$$\Delta_- f(i,j) = f(i, j+1) - f(i+1, j)$$

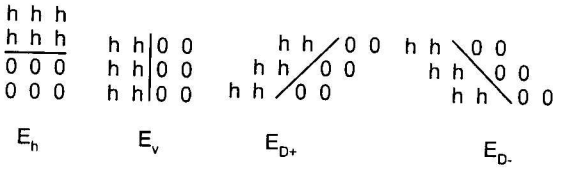


Table of differences for an edge

	E_h	E_v	E_{D+}	E_{D-}
$\lvert\Delta_{2x}\rvert$	0	h	h	h
$\lvert\Delta_{2y}\rvert$	h	0	h	h
$\lvert\Delta_{+}\rvert$	h	h	0	h
$\lvert\Delta_{-}\rvert$	h	h	h	0
$\sqrt{\Delta_{2x}^2 + \Delta_{2y}^2}$	h	h	h√2	h√2
$\sqrt{\Delta_{+}^2 + \Delta_{-}^2}$	h√2	h√2	h	h

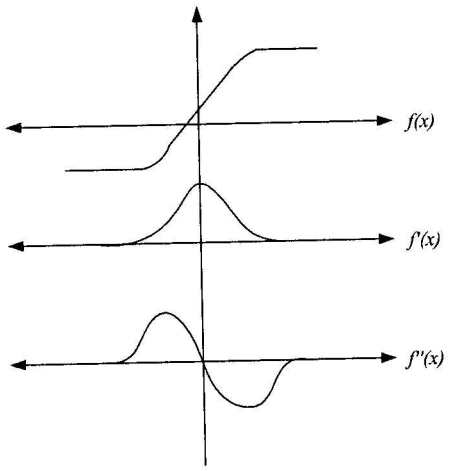
The last two entries show the bias for/against the diagonal edges.

Laplacian operator

$\Delta_{xx} f + \Delta_{yy} f$

$$\left|\nabla^2 f\right| = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$Mask(Laplacian) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



The location of the edge corresponds to the zero crossing of the laplacian (2^{nd} Derivative)

Note: The derivatives of the image are "very" sensitive to noise, so we smooth image first.

Convolve an image with the Laplacian of a 2-D Gaussian function of the form:

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{r^2}{2\sigma^2}}$$

$$\nabla^2(h * f) = \nabla^2 h * f$$

$$\nabla^2 h = \frac{r^2 - \sigma^2}{\sigma^4} e^{-\frac{r^2}{2\sigma^2}}$$

Note:

$r^2 = x^2 + y^2$

because

$x = r\cos\theta$

$y = r\sin\theta$

The actual solution of this is consider trivial and is not included

Prewitt Edge Operator

$$Mask\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad Mask\Delta_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobal Edge Operator

$$Mask\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Mask\Delta_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Example of Sobel operator,

$$\mathrm{Im} = \begin{bmatrix} 0 & 10 & 10 \\ 1 & 10 & 11 \\ 1 & 10 & 12 \end{bmatrix}$$

$$Mask\,\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$Mask\,\Delta_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\Delta_x f = (1*10 + 2*11 + 1*12) + (-1*0 + -2*1 + -1*1) = 41$$

$$\Delta_y f = 3$$

$$Gradient\ \nabla f = \sqrt{(\Delta_x f)^2 + (\Delta_y f)^2} = \sqrt{41^2 + 3^2}$$

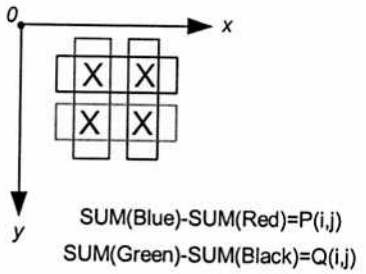
Laplacian Edge Dectectors

$$Mask_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$Mask_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 10.10 of house with different examples of edge detection
(Edges that are detected become "highlighted")
Figure 10.11 is of the same house but smoothed first then edge detection applied
Figure 10.12 uses sobel operator
Figure 10.13 Laplacian Operator
Figure 10.14 Laplacian of a Gaussian (LoG)
$\nabla^2(h * f) = \nabla^2 h * f$
Figure 10.15 results of LoG
Figure 10.16

Canny Edge Detector



$$S(i,j) = G[i,j,\sigma] * I[i,j]$$

The magnitude and orientation of the gradient are:

$$M(i,j) = \sqrt{P^2 + Q^2}$$

$$\theta(i,j) = \tan^{-1}\left(\frac{Q}{P}\right)$$

Hough Transform (Detecting curves of given shapes)

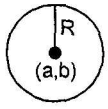
Simple example: Is there an edge/line passing through point p? if yes find its slope.

$$m = \frac{y_i}{x_i}$$

Hough Transform

Hough transform is applied to an image to detect lines, shapes, etc by taking an edge image and mapping it to parameter spce (e.g. (ρ,θ)) space. In implementation we normally have (ρ,θ) space as an accumulator matrix and the points with the highest values (highest intensities when treated as an image) provide information about the object. For each type of object you are looking for, the Hough transform is applied using a different mapping.

e.g., Detection of circles with a given radius R



Circle Equation: $(x-a)^2+(y-b)^2=R^2$

To apply this method we first select the expected ranges of values for a and b for the circle, giving us $(a_{\max},a_{\min})$ and $(b_{\max},b_{\min})$. This allows us to create a parameter space (a,b) that is used as an accumulator matrix. We next take each point $p(x_i,y_i)$ and consider all possible values of (a, b) in the parameter space. For each solution to the equation, the corresponding accumulator $A(a_i,b_j)$ is incremented by 1. When all points within the image space have been applied, the points in the accumulator with the highest values are the probable locations of the centers of circles.

Line Detection

There are two methods that can be applied, the first is based upon the slope-intercept equation and the second is based upon the (ρ,θ) equation.

a)Slope-Intercept method

$y = mx + b$

b)Sinusoidal method

$x\cos\theta + y\sin\theta = \rho$

The parameter space (ρ,θ) is digitized, then for each point $p(x_i, y_i)$, we solve the sinusoidal equation for possible solution values for (ρ_i,θ_j), then we increment the accumulator A (the parameter space location), $A(\rho_i,\theta_j)$. Once again the locations with the highest values determine the parameters of the line.

Representation and Description:

Chain Codes

The grid size determines the number of codes used to represent an image. Smaller grid size provides finer chain code representation.

Rotation of an object by a multiple of 90 degrees (i.e. n*90), can be obtained by adding n mod 4 to the original code.

Difference code is rotation invariant (If the object is rotated the difference code will still be the same)
Difference = Current – Previous (using wrap around)

Code (original) (circular code)	1	0	1	0	3	3	2	2
Code (Rotated by 90), i.e. add 1 to each code	2	1	2	1	0	0	3	3
Difference Code in both original and rotated cases	3	3	1	3	3	0	3	0

Shape Numbers

Te shape number of a boundary based on the 4-directional codes is defined as the first difference chain code of smallest magnitude. (Find the difference code and then find the lowest code starting from the left. For the above example.

Difference Code	3	3	1	3	3	0	3	0
Shape Number	0	3	0	3	3	1	3	3

Shape Complexity

$Complexity = \frac{p^2}{4\pi A}$ where p is the boundary and A is the area of the image.

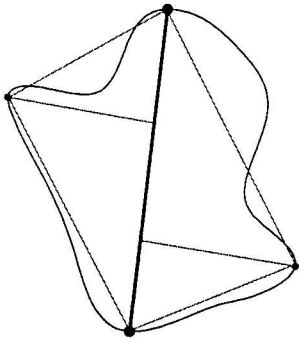
It exploits deviation from convexity

For circle: $\frac{4\pi^2 r^2}{4\pi^2 r^2} = 1$

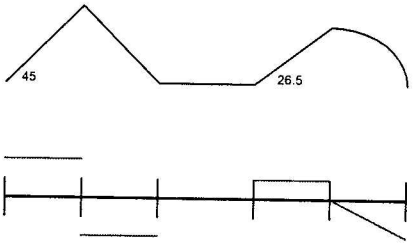
For a square with edge a: $\frac{(4a)^2}{4\pi a^2} = \frac{4}{\pi}$

Polyline Representation

For a closed boundary, start with two farthest points in the boundary, then find the distance of the boundary point farthest from the line joining its two end points. If this is greater than a threshold, make it a vertex.



Slope Representation

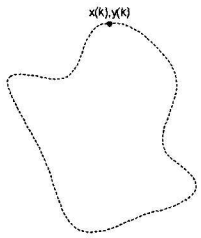


Signature

Signature is a 1-D representation of a boundary. (i.e. The distance from the centroid to the boundary.)

This representation is invariant to translation, but its signature changes with rotation and scaling.
To make signature invariant to rotation, select starting point to be the point farthest from the centroid.

Fourier Descriptors



A string of numbers representing the contour, $S(k) = [x(k), y(k)], k = 0,1,...,N-1$
We can represent each coordinate pair as a complex number.
$S(k) = x(k) + j \cdot y(k)$, Find its DFT (Discrete Fourier Transform).

DFT: $a(u) = \frac{1}{N}\sum_{k=0}^{N-1} S(k) e^{-\frac{j2\pi}{N}ku}, u = 0,1,2,...,N-1$

IDFT: $S(k) = \sum_{u=0}^{N-1} a(u) e^{j\frac{2\pi}{N}uk}, k = 0,1,2,...N-1$

Now, if we use only M coefficients: $S'(k) = \sum_{u=0}^{M-1} a(u) e^{j\frac{2\pi}{N}uk}, k = 0,1,2,...,N-1$, we obtain an approximate,

Note for $x(n) \leftrightarrow X(k)$

$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk}, k = 0,1,2,...,N-1$

The more coefficients you use, the more refined the object's boundary will be (refer to the circle/square change in book).

Sensitivity of Fourier Descriptors

1. To Rotation

$$S_r(u) = S(k)e^{j\theta}$$

$$a_r(u) = \frac{e^{j\theta}}{N}\sum_{n=0}^{N-1} S(k)e^{-j\frac{2\pi}{N}uk}$$

giving

$$a_r(u) = e^{j\theta}a(u)$$

2. To Translation

$$S_t(k) = S(k) + \Delta_{xy}$$

$$a_t(u) = \frac{1}{N}\sum_{n=0}^{N-1}(S(k) + \Delta_{xy})e^{-j\frac{2\pi}{N}nk} = \frac{1}{N}\sum_{n=0}^{N-1} S(k)e^{-j\frac{2\pi}{N}nk} + \Delta_{xy}\frac{1}{N}\sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}nk}$$

giving

$$a_t(u) = a(u) + \Delta_{xy}\delta(u)$$

because

$$\sum_{k=0}^{N-1} x^k = \frac{1-x^N}{1-x}$$

3. To Scaling

$$S_s(k) = \alpha S(k)$$

$$a_s(u) = \alpha a(u)$$

Region Descriptors

1. Topological
   a. Euler Number E
      i. $E = C - H$
         1. C is the number of connected components
         2. H is the no. of holes

2. Statistical Approaches
   a. Based on first order statistics (z_i is a random variable representing gray level intensity)
      i. Mean $m = \sum_{i=1}^{L} z_i p(z_i)$
         1. Drawback – A square image with ½ black and ½ white would have the same mean as a black/white checkerboard image
      ii. Moments $\mu_n = \sum_{i=1}^{L} (z_i - m)^n p(z_i)$
         1. Note, for $n = 2$,this function gives you the variance.

   b. Co-occurrence Matrix
      i. It is based on 2nd order statistics describing how often the possible pairs of gray levels occurs.
      ii. Example:

$$f = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 2 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 \end{bmatrix}$$

To get the co-occurrence matrix C,
Need a Direction vector, e.g., $\delta = (1,0)$ where (1,0) is (x,y), so we would be looking for co-occurrence of 0 0, next to each other looking to the right. Similarly for 0 1, etc.. For the above image, we have the following co-occurrence:

$$C = \begin{bmatrix} 0 & 3 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Since there are a total possible pairs of 12, we divide all elements by 12 to find the probability measure.

$$C = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{6} & \frac{1}{4} & \frac{1}{12} \\ 0 & \frac{1}{12} & \frac{1}{6} \end{bmatrix}$$

iii. So the elements of the co-occurrence matrix C is defined, $C_{i,j}(\delta) = \frac{M_{i,j}(\delta)}{N(\delta)}$ where

$M_{i,j}$ is the number of times that a point having gray level z_i occurs in position δ relative to a point having gray level z_i.

N(δ) is the number of point pairs in f in relative position δ.

δ can be any values so δ = (-2,0) would be looking from the current point to the point two positions to the left. It can easily be a diagonal line of any slope too.

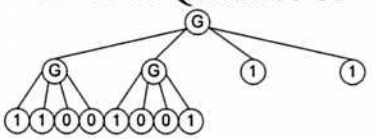
3. Quadtree
   a. Example

1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Format of Quadrants

I	II
III	IV

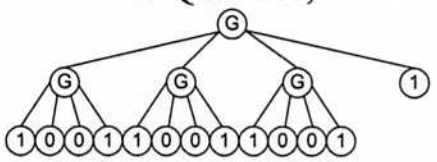
Gives us Quadtree of



   b. Example

1	0	1	0
0	1	0	1
1	0	1	1
0	1	1	1

Its Quadtree,



Maximal Blocks (Medial Axis Transformation)
The region can be represented by a set of maximal blocks (a set of centers P, radii r_p, and values V_p). The maximal block at point p is the largest square of uniform values centered at p such that it is not contained in any other maximal block.

Examples

Image:

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

based on the underlined center we achieve a maximal block of (x,y,r)=(3,3,2)

Image: In this example we are listed "all possible" maximal blocks

1	1	1	1	1
1	1	1	1	0
1	1	1	1	1
1	1	1	1	1
0	1	1	1	1

x	y	r
5	1	0
2	2	1
3	2	1
4	4	1
2	3	1
3	3	1
3	4	1

Image:

1	1	1	1	1
1	1	1	1	1
1	1	0	1	1
1	1	1	1	1
1	1	1	1	1

In this case each item is a maximal block, so there are 24 maximal blocks. This representation is very sensitive to noise.

Geometric Properties of Regions

- Area: This is the total number of pixels the image contains.
  - $A = \sum_{i=1}^{N} \sum_{j=1}^{M} B(i,j)$
    - *B(i,j) is the binary value at position(i,j) of the region.*

- Perimeter: This is the total number of pixels on the boundary of the object.

- Position (Centroid)

$$\bar{x} = \frac{\sum_{x=0}^{M}\sum_{y=0}^{N} xf(x,y)}{\sum_{x=0}^{M}\sum_{y=0}^{N} f(x,y)}$$

  - $$\bar{y} = \frac{\sum_{x=0}^{M}\sum_{y=0}^{N} yf(x,y)}{\sum_{x=0}^{M}\sum_{y=0}^{N} f(x,y)}$$

Example

2	1	1
3	1	0
3	2	1

$$\bar{x} = \frac{3(0)+3(0)+2(0)+2(1)+1(1)+1(1)+1(2)+0(2)+1(2)}{14} = \frac{8}{14}$$

$$\bar{y} = \frac{3(0)+2(0)+1(0)+3(1)+1(1)+0(1)+2(2)+1(2)+1(2)}{14} = \frac{12}{14}$$

- Moments
  - $m_{i,j} = \sum_x \sum_y x^i y^j f(x,y)$
  -

i	j	$m_{i,j}$
0	0	14
1	0	8
0	1	12
2	0	12
1	1	7
0	2	20

$$m_{0,0} = \sum_x \sum_y f(x,y)$$

$$m_{1,0} = \sum_x \sum_y xf(x,y)$$

$$m_{0,1} = \sum_x \sum_y yf(x,y)$$

$$m_{1,1} = \sum_x \sum_y xyf(x,y)$$
$$m_{2,0} = \sum_x \sum_y x^2 f(x,y)$$

$m_{0,0}$ The area of a region

$m_{0,2}$ The moment of inertia of f around x axis

$m_{2,0}$ The moment of inertia of f around y axis

m_0 The moment of inertia about the origin → $\sum_x \sum_y (x^2 + y^2) f(x,y) = m_{0,2} + m_{2,0}$

Centroid (using Moments)

$$\bar{x} = \frac{\sum_{x=0}^{M} \sum_{y=0}^{N} xf(x,y)}{\sum_{x=0}^{M} \sum_{y=0}^{N} f(x,y)} = \frac{m_{1,0}}{m_{0,0}}$$

$$\bar{y} = \frac{\sum_{x=0}^{M} \sum_{y=0}^{N} yf(x,y)}{\sum_{x=0}^{M} \sum_{y=0}^{N} f(x,y)} = \frac{m_{0,1}}{m_{0,0}}$$

Central Moments

$$\mu_{i,j} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j f(x,y)$$

$$\mu_{0,0} = m_{0,0}$$

$$\mu_{1,0} = \sum_x \sum_y (x - \bar{x}) f(x,y) = m_{1,0} - m_{0,1} = 0$$

$$similarly, \mu_{0,1} = 0$$

$$\mu_{2,0} = m_{2,0} - \frac{m_{1,0}^2}{m_{0,0}}$$

Note these derivations are done through algebraically adjusting and changing the summation notation and then substituting the moments for things known.

Normalized Central Moments $\eta_{p,q}$

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{\gamma}}, \text{ where } \gamma = \frac{p+q}{2} + 1$$

Define a set of 7 invariant moments as follows:

$\phi_1 = \eta_{2,0} + \eta_{0,2}$

$\phi_2 = (\eta_{2,0} - \eta_{0,2})^2 + 4\eta_{1,1}^2$

.

.

(See the text)

When completely solved, all 7 invariant moments will give you a 1x7 column "feature" vector.

- Affine Transform

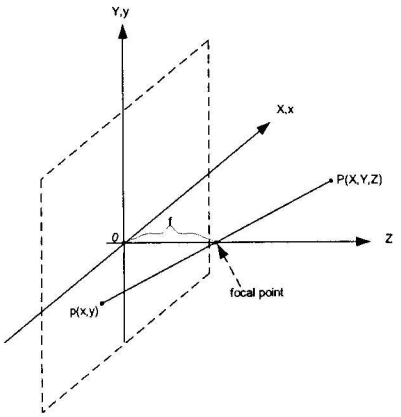
  $x^* = ax + b$

  $y^* = cy + d$

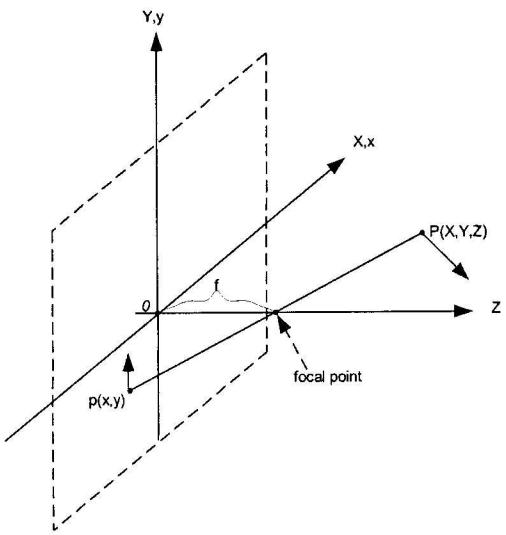
  If $a = c = 1$, basically you are translating

Optical Flow

When an object moves in three-dimensional space, the motion of its image in the image plane produces a flow field called *optical flow*. Optical Flow can be used to segment an image into regions that correspond to different objects, or to study the motion of objects in a three-dimensional scene.

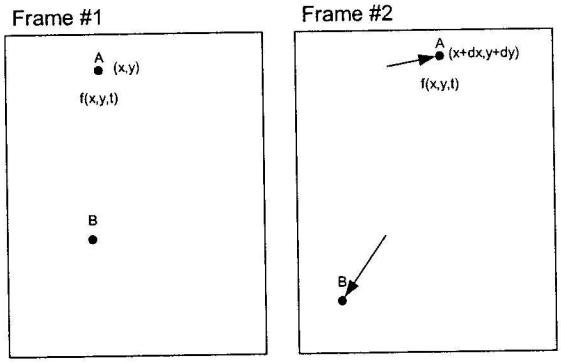


When an object moves,



Calculation of Optical Flow

Consider image intensity as a function of position and time, $f(x,y,t)$. The image at time $t+dt$ is the result of original image at time t being moved by (dx,dy), i.e. $f(x+dx,y+dy,t+dt)=f(x,y,t)$



Now, using Taylor's expansion

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt + \cdots$$

Considering that $f(x+dx, y+dy, t+dt) = f(x,y,t)$, we can write

$$\frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt = 0$$

$$\frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t}\frac{dt}{dt} = 0$$

$\frac{dx}{dt} = u$ is the velocity in the x-direction

$\frac{dy}{dt} = v$ is the velocity in the x-direction

With this notation we can rewrite the previous equation as:

$$\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v = -\frac{\partial f}{\partial t} \quad (\textit{optical flow constraint equation})$$

In Practice,

$$\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v \neq -\frac{\partial f}{\partial t}$$

We define an error function

$$\varepsilon_b = f_x u + f_y v + f_t$$

Also, we define a smoothness constraint

$$\varepsilon_s^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

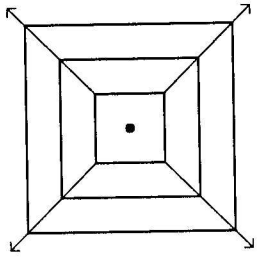
Therefore, we minimize

$$\varepsilon^2 = \iint \left(\alpha \varepsilon_s^2 + \varepsilon_b^2\right) dxdy$$
$$\varepsilon^2 = \iint \left[\alpha^2\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) + f_x^2 u^2 + f_y^2 v^2 + f_t^2 + 2f_x f_y uv + 2f_x f_t u + 2f_x f_t v\right] dxdy$$

This can be solved using calculus of variations.

Focus of Expansion

If an object point is in translational motion, the motion of its image point in the image plane produces a flow which passes through a specific point called the *focus of expansion*.



As the square of pixels moves towards us it is creating a flow, in which, all points of the square pass through the *FOE* at the center.

Consider the following:

$$x = \frac{fX}{Z - f}$$

$$y = \frac{fY}{Z - f}$$

Let $f = 1, z >> 1$, so

$$x = \frac{X}{Z}$$

$$y = \frac{Y}{Z}$$

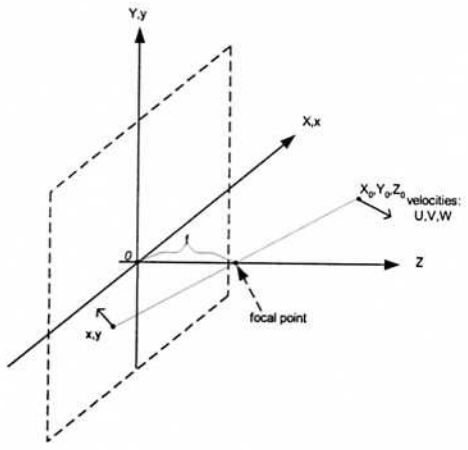
Consider an object point (X_0, Y_0, Z_0) moving with velocities

$$(U, V, W) = \left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{dZ}{dt}\right)$$

We can write

$$x(t) = \frac{X(t)}{Z(t)} = \frac{X_0 + Ut}{Z_0 + Wt}$$

$$y(t) = \frac{Y(t)}{Z(t)} = \frac{Y_0 + Vt}{Z_0 + Wt}$$



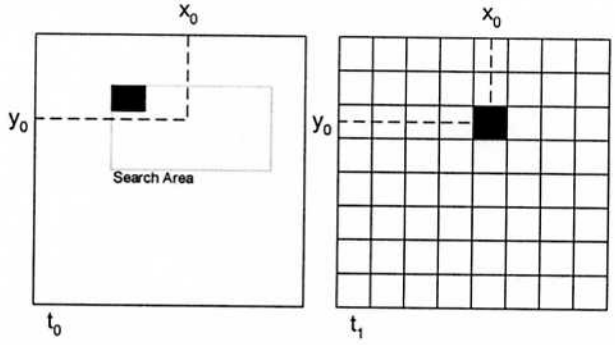
Focus of Expansion is:

$$x|_{t\to-\infty}=\frac{U}{W}$$

$$y|_{t\to-\infty}=\frac{V}{W}$$

i.e., a set of points with the same velocity, all pass through the same point *FOE*.

Common Method of Motion Estimation



To find the corresponding block, we search the search area until we find the minimal *Mean Absolute Difference*. This allows us to find where the object has moved.

Match Measure: $\sum\sum|f(x,y,t)-f(x+dx,y+dy,t+dt|$

Minimum Distance Classifier

Suppose each pattern class is represented by a prototype (mean) vector m_j.

$$m_j = \frac{1}{N_j} \sum_{x \in \omega_j} x$$

To classify x, an unknown pattern, we assume the distance,

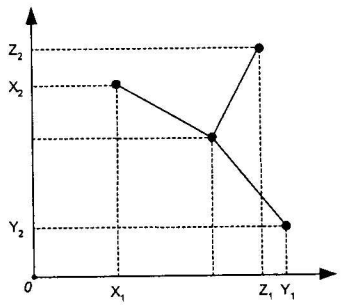
$$D_j(x) = \|x - m_j\|, j = 1,2,\cdots,M$$

Note that $\|a\| = (a^T a)^{\frac{1}{2}}$, therefore

$$\begin{aligned} D_j^2(x) &= \|x - m_j\|^2 \\ &= (x - m_j)^T (x - m_j) \\ &= (x^T - m_j^T)(x - m_j) \\ &= x^T x - x^T m_j - m_j^T x - m_j^T m_j \\ &= x^T x - 2\left(x^T m_j - \frac{1}{2} m_j^T m_j \right) \end{aligned}$$

So, the decision function is

$$d_j(x) = x^T m_j - \frac{1}{2} m_j m_j^T$$



Term Projects Discussed → Groups of 3
Procedure involves,

- Presentation
- Short paper (much like journal paper)
    - Due last week of class (group provides 1 write up on disk & hardcopy)

Examples of Project Topics:

- Image Enhancement/Filtering
- Biomedical Image Processing/Analysis
- Applied Machine Vision
    - Fingerprints
    - Face Recognition
    - Security Topics
- Image Restoration
- Other topics must be approved.

    - Journals that can be used for this
        - Signal Processing
        - Machine Vision & Applications
        - Image Computing
        - Optical Engineering (Hardcopy)
        - Electronic Imaging (Hardcopy)
        - International Journal of Imaging Systems
        - Journal of Visual Communication & Image Representation
        - Real Time Imaging
        - ACM/IEEE Proceedings